Hardware-Accelerated Dynamic Volume Rendering for Real–Time Surgical Simulation

Marco Agus, Andrea Giachetti, Enrico Gobbetti
Gianluigi Zanetti, and Antonio Zorcolo

CRS4
Center for Advanced Studies, Research and Development in Sardinia
POLARIS Edificio 1, 09010 Pula (CA), Italy
{magus,zarco,giach,gobbetti,zag}@crs4.it

Abstract. We developed a direct volume rendering technique, that supports low latency real time visual feedback in parallel with physical simulation on commodity graphics platforms. In our approach, a fast approximation of the diffuse shading equation is computed on the fly by the graphics pipe-line directly from the scalar data. We do this by exploiting the possibilities offered by multi-texturing with the register combiner OpenGL extension, that provides a configurable means to determine per-pixel fragment coloring. The effectiveness of our approach, that supports a full decoupling of simulation and rendering, is demonstrated in a training system for temporal bone surgery.

1 Introduction

The diffusion of minimally invasive procedures is bringing major improvements in the quality of the care provided to the patients. This is done, however, at the cost of the increase in the complexity of the surgical procedures performed, thus requiring an increase in the training needed by each specific intervention for both planning and procedural issues. At the same time, the shortage of cadavers for medical training and public concern with the inhuman treatment of animals is drastically limiting the traditional approaches to surgical training. Virtual reality simulators realistically mimicking a patient-specific operating environment would therefore significantly contribute to the improvement of surgical training. Developing high quality simulators is, however, extremely difficult, since the need to provide real-time feedback to users, while simulating physical effects, imposes stringent constraints on the simulation and visualization system. This paper focuses on the dynamic visualization problem, describing the volume rendering technique that we have developed for a temporal bone dissection simulator.

Direct volume rendering with shading, that works by integrating along selected projectors the value of a continuous emission/reflection/absorption volume function reconstructed from discrete sampling points [1], is the de-facto standard in the pre-operative analysis of medical data. By manipulating the mapping from values of the original volume data to emission, reflection, and absorption coefficients, various effects can be achieved, including isosurfaces and opaque objects. Using this data intensive technique on dynamic volumes under real-time constraints is, however, an open research problem.

This fact has limited simulators to employ surface based techniques, that have problems with semitransparent materials and rely on complex mesh structures that impose important synchronization overheads. A number of authors have proposed to exploit texture mapping and rasterization hardware to render scalar volumes at interactive speeds [2–5]. These techniques are based on uploading the scalar volume to texture memory prior to rendering object-aligned or view-direction-aligned textured volume slices. One of the major limitations of these methods is their inability to efficiently implement surface illumination models, since texture lookup is based only on data values and not on gradient information. The quality of the images is therefore insufficient for accurately perceiving surface shape. Various authors have thus proposed alternative techniques for supporting hardware-accelerated direct volume rendering with shading [4, 6–8]. While image quality is close to that of the best software solution, this comes at the expense of performance and texture memory overheads, since the proposed techniques require multiple passes through the rasterization hardware and/or precomputation of gradient volumes. This is unacceptable in surgical simulation, since the volume is continuously varying, and thus we cannot efficiently compute and reload gradient maps without strongly coupling the simulation and rendering tasks.

In this paper, we propose a texture-based volume rendering approach, that supports low latency real time visual feedback to occurr in parallel with physical simulation, without requiring any synchronization among the threads. In our approach, a fast approximation of the diffuse shading equation is computed on the fly by the graphics pipe-line directly from the scalar data. We do this by exploiting the possibilities offered by multi-texturing with the register combiner OpenGL extension, that provides a configurable means to determine per-pixel fragment coloring. The rest of the paper describes our technique and illustrates its effectiveness in a virtual training system for temporal bone surgery [9].

2 Interactive volume rendering approach

Although volumetric data is defined over a continuous three-dimensional domain (R^3), measurements and simulations provide volume data as 3D arrays, that can be easily used as scalar texture images, without pre-processing. We render this volume sampling the volume through texturing hardware using front-to-back slice composition. For each step and for all pixels, graphics hardware accesses the texture and extracts the scalar value. This sampled value is converted, through a transfer function, to a color triple and an opacity value that are saved inside combiner input registers. Combiners are then programmed to compute gradients and shading on-the-fly, and return the color used during the blending process. In the following, we provide more details about this process.

Sampling through texture mapping. Current consumer graphics hardware is based on an object-order rasterization approach, i.e. primitives (polygons, lines, points) are scan-converted and written pixel-per-pixel into the frame buffer. Since volume data do not consist of such primitives, a proxy geometry is defined for each individual slice through volume data. Each slice is textured with corresponding data from tvolume. The

volume is reconstructed during rasterization on slice polygons by applying a convolution of volume data with a filter kernel. The entire volume can be represented by a stack of such slices, if the number of slices satisfies restrictions imposed by Nyquist theorem. Our approach follows the technique proposed by Rezk-Salama and others [7], extending it with on-the-fly gradient and shading computation. In our case, since in a surgical setting viewpoint motion is constrained (limited to maximum 30 degrees), we are allowed to use object-aligned slices. At each frame, we traverse the array of slices, and we reload them as textures two at a time. After texture loading and reconstruction, the rasterization process derives, for each projected pixel, texture sampling position, and texturing hardware extracts the correspondent density, by bi-linear or tri-linear interpolation of closest texels; the resultant value is then mapped to an RGBA vector by the transfer function. In order to compute the surface gradient, four texture units are needed, that are used to sample the volume with offset dx, dy, dz, relatively to the central point.

This procedure is extremely efficient, since all the computation is performed in parallel in the graphics hardware and no particular synchronization is needed between the renderer and the process that is modifying the dataset. Only a single sweep through the volume is needed, and volume slices are sequentially loaded into texture memory on current standard PC graphics platform using AGP 8X transfers, which provides a peak bandwidth of 2108 MB/s.

Transfer function. The transfer function mapping, in our direct volume rendering approach, is obtained by exploiting the *glColorTable* primitive, that is commonly implemented in commodity graphics hardware. This function is used by compiling a look-up table, with transfer function values, and by installing it inside graphics memory. At the same time of sampling, texturing hardware performs transfer conversion of density values to RGBA colors contained inside the table. The color table contains associated colors instead of pure colors [10], in order to control the color interpolation error. The associated color employment has also beneficial effects to color accumulation processs. The color look-up table lets users choose and calibrate the transfer function in real time; it can be computed and reloaded each time the user change some function parameters.

Voxel color computation. Lighting and shading process follows the standard lighting equation [10]:

$$\widetilde{C}[n] = \alpha[n]C_a[n]l_a + \alpha[n]C_d[n]\left\|\bar{l}_d\right\| max\left(\frac{\nabla f[n]}{\|\nabla f[n]\|}\cdot\frac{\bar{l}_d}{\left\|\bar{l}_d\right\|}, 0\right) \qquad (1)$$

where $\widetilde{C}[n]$ is the associated color, $C_a[n]$, $C_d[n]$ and $\alpha[n]$ are the non-directional ambient reflective factor, the diffuse directional reflective factor and the voxel opacity, while l_a and $\bar{l}_d$ are the ambient light intensity and the light intensity coming from the directional source. In order to highlight surface details, we employ the artifact proposed by [11] of weighting the opacity $\alpha[n]$ with a surface *strength*, evaluated as a function of volume and his gradient: $S = h\left(f(s), \nabla f(s)\right)$ [12]. If we use the gradient modulus as *strength*, we have:

$$\widetilde{C}[n] = l_a \left\|\nabla f[n]\right\| \alpha[n] C_a[n] + \left\|\bar{l}_d\right\| \alpha[n] C_d[n] max\left(\nabla f[n] \cdot \frac{\bar{l}_d}{\left\|\bar{l}_d\right\|}, 0\right). \quad (2)$$

Such a strength function, enables the visualization of boundary surface between tissues, and disables the visualization of parts with null gradient (like the internal parts of an object). Specifically, in our case, the shading components are supposed to be the combination of an ambient component and a directional component emitted by a source oriented along the volume z axis (slices normal). This way, the dot product between the light direction and the opacity gradient is the component $\nabla_z f[n]$, and equation 1 is simplified as follows:

$$\widetilde{C}[n] = l_a \left\|\nabla f[n]\right\| \alpha[n] C_a[n] + l_d \alpha[n] C_d[n] max\left(\nabla_z f[n], 0\right). \quad (3)$$

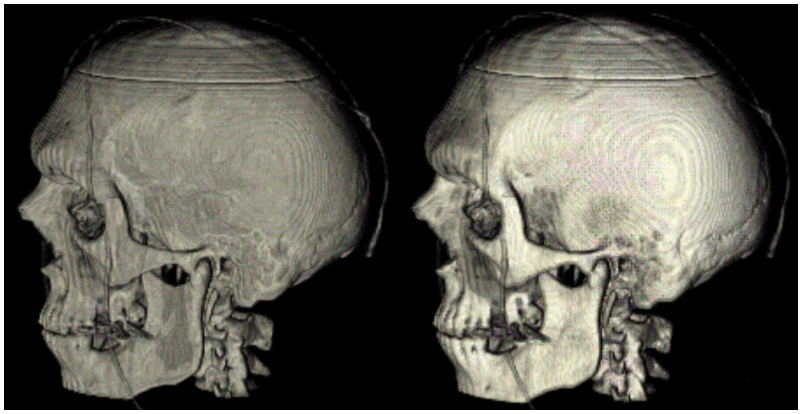

Fig. 1. Internal and external optical models.

But visual result of equation 3 is not fully satisfying: in fact only surface voxels contribute to pixel color, because strength becomes zero in tissue internal parts. This fact would be irrelevant if surfaces were consistent enough to completely mask the color of internal voxels. Anyway, low strength surfaces and small width walls let see the hollows produced by equation 3 (see fig. 1a). Hence, the optical model employed for internal volumes is different from that used exclusively for parts having non null gradient. The overall model is then defined by:

$$\widetilde{C}[n] = \begin{cases} l_a \left\|\nabla f[n]\right\| \alpha[n] C_a[n] + l_d \alpha[n] C_d[n] max\left(\nabla_z f[n], 0\right) & if \ \left\|\nabla f[n]\right\| > 0 \\ \left(l_a + l_d\right) \alpha[n] C_a[n] & if \ \left\|\nabla f[n]\right\| = 0 \end{cases}. \quad (4)$$

With this artifact, the image quality is greatly improved during rendering of low density tissues or in these case of tissue subtle layers with high density (for example bone), as shown in figure 1b.

Opacity gradient computation. In lighting equation 2, the surface normal is related to gradient $\nabla f[n]$, and the modulus is regarded as surface *strength*. If $f[n]$ is used as opacity, instead of density, we can arbitrarily modify the surface appearance properties (opacity, width and consistence) by modifying the transfer function. Since OpenGL register combiners receive from texture hardware 4 opacity values $\alpha(p)$, $\alpha(p + dx)$, $\alpha(p + dy)$, $\alpha(p + dz)$, they are able to approximate the opacity gradient with forward differences. Since combiners are SIMD arithmetic modules able to perform linear operations, it is relatively simple to derive forward differences and vector modules, but it is impossible to perform ratios and root extractions. Now the gradient norm computation involves a square root computation, that needs to be approximated with a polynomial function. Since the number of available combiners is limited and many of them are used to compute the gradient components as well as the lighting equation, we can only approximate the square root function with a quadratic function. The 2nd order polynomial is derived from a Taylor series evaluated in the neighborhood of an arbitrary point x_0 of interval]0, 1]. The value of x_0 has to be choosen in order to minimize the approximation error in the interval. According to equation 2, the gradient norm is used to weight voxel opacity and associated color contribution, so the best approximation is obtained when Taylor series is evaluated in $x_0 = 1$. The interpolation function is then $\sqrt{x} \approx \frac{3}{8} + \frac{3}{4}x - \frac{1}{8}x^2$.

Performance enhancement. Pixel fill-rate is the major limiting factor when using a texturing approach to volume visualization. In zoom rendering, an appropriately downscaled image is rendered in the back buffer and then enlarged and copied to the front buffer [13, 14]. In this way, delays associated with buffer swap synchronization are avoided, and the number of pixels filled during volume rendering is reduced. In our implementation, the copy and zoom operations are implemented by copying the reduced size image in texture memory and then rendering a textured polygon in the front buffer. Hence, sophisticated texture interpolation algorithms can be used to reduce the artifacts caused by magnification.

3 Implementation and results

Our technique for direct volume rendering has been integrated in a prototype training system for mastoidectomy. The simulator system provides real–time visual and haptic feedback [9, 15] and it is modeled as a collection of loosely coupled concurrent components [16]. The overall system is divided in a "fast" subsystem, responsible for the high frequency tasks (surgical instrument tracking, force feedback computation, bone erosion), and a "slow" one, essentially dedicated to the production of data for visual feedback [17]. The system runs on two interconnected multiprocessor machines. Thanks to our volume rendering approach, the renderer is totally decoupled from the simulator

and the tracking system, and runs at his own frequency. The current configuration is the following: a single-processor PIV/1500 MHz with 256 MB PC133 RAM for the high-frequency tasks (haptics loop (1KHz) and interprocess communication loop); a dual-processor Intel Xeon 2.4 GHz with 2048 MB DDR PC400 RAM and a NVIDIA GeForce FX 5800 Ultra AGP 8X and running a 2.4 linux kernel, for the low frequency tasks (receiving loop, simulator evolution and visual rendering); a Phantom Desktop and a Phantom 1.0 haptic devices, that provide 6DOF tracking and 3DOF force feedback for the burr/irrigator and the sucker; a n-vision VB30 binocular display for presenting images to the user. We are currently using a volume of 256x256x128 cubical voxels (0.3 mm side) to represent the region where the operation takes place. We executed performance benchmarks on the system, which revealed that, using eight register combiners and 8 bit/texel volumes, peak texture transfer rate is about 400M texel/s, while peak fill rate is about 400M pixel/s per second. According to these results, the rendering system should theoretically be able to completely reload and render an entire $256X256X128$ dataset in about 40 ms per frame. In the surgical simulator system, with this volume size, and using a window of about the same resolution ($320X240$ zoomed to $640X480$), we obtain refresh timings of about 50 msec per frame, corresponding to a frame rate of 20 fps, which is close to the theoretical peak. The CPU overhead is negligible, and the simulation can run and update the volume in parallel in a totally decoupled manner. The performance of the prototype is thus sufficient to meet timing constraints, even though the computational and visualization platform is constructed from affordable and widely accessible components. The visual quality of the method is illustrated in figure 2, which shows snapshots captured during a virtual session of the surgical simulator. The principal steps of a basic mastoidectomy, performed by an Ear, Nose and Throat surgeon, are represented.

4 Conclusion and discussion

We presented a dynamic volume rendering technique which is well suited for the incorporation in surgical simulators. The technique supports low-latency and high frequency rendering of shaded semi-transparent materials. The method is extremely efficient, since all the computation in performed in parallel in the graphics hardware and no particular synchronization is needed between the renderer and the process that is modifying the dataset. Only a single sweep through the volume is needed, and volume slices are sequentially loaded into texture memory on current standard PC graphics platform using AGP transfers. The effectiveness of our approach is demonstrated in a training system for temporal bone surgery.

Acknowledgments

We would like to thank Prof. Stefano Sellari Franceschini and his team, University of Pisa, for his collaboration in the design and testing of the system.

References

1. Max, N.: Optical models for direct volume rendering. IEEE Transactions on Visualization and Computer Graphics **1** (1995) 99–108
2. Cabral, B., Cam, N., Foran, J.: Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In Kaufman, A., Krueger, W., eds.: 1994 Symposium on Volume Visualization, ACM SIGGRAPH (1994) 91–98 ISBN 0-89791-741-3.
3. Guan, S., Lipes, R.G.: Innovative volume rendering using 3D texture mapping. In: Image Capture, Formatting and Display. Volume 2164 of SPIE. SPIE (1994)
4. Van Gelder, A., Kim, K.: Direct volume rendering with shading via three-dimensional textures. In: 1996 Volume Visualization Symposium, IEEE (1996) 23–30 ISBN 0-89791-741-3.
5. Kulick, T.: Building an opengl volume renderer. SGI Dev. News (1996)
6. Westermann, R., Ertl, T.: Efficiently using graphics hardware in volume rendering applications. In Cohen, M., ed.: SIGGRAPH 98 Conference Proceedings. Annual Conference Series, ACM SIGGRAPH, Addison Wesley (1998) 169–178 ISBN 0-89791-999-8.
7. Rezk-Salama, C., Engel, K., Bauer, M., Greiner, G., Ertl, T.: Interactive volume rendering on standard PC graphics hardware using multi-textures and multi-stage rasterization. In Spencer, S.N., ed.: Proceedings of the 2000 SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware, ACM Press (2000) 109–118
8. Engel, K., Kraus, M., Ertl, T.: High quality pre-integrated volume rendering using hardware-accelerated pixel shading. In: EuroGraphics/SIGGRAPH Workshop on Graphics Hardware. (2001)
9. Agus, M., Giachetti, A., Gobbetti, E., Zanetti, G., Zorcolo, A.: Adaptive techniques for real time haptic and visual simulation of bone dissection. In: IEEE Virtual Reality Conference, Conference held in Los Angeles, CA, USA, March 22–26 (2003) 102 –109
10. Wittenbrink, C.M., Malzbender, T., Goss, M.E.: Opacity-weighted color interpolation for volume sampling. In: IEEE Symposium on Volume Visualization, IEEE, ACM SIGGRAPH (1998) 135–142
11. Levoy, M.: Display of surfaces from volume data. IEEE Computer Graphics and Applications **8** (1988) 29–37
12. Drebin, B., Carpenter, L., Hanrahan, P.: Volume rendering. In Wolfe, R., ed.: Significant Seminal Papers of Computer Graphics: Pioneering Efforts that shaped the Field, N.Y., ACM Press (1998) 363–372
13. Mazuryk, T., Schmalstieg, D., Gervautz, M.: Zoom rendering: Improving 3-D rendering performance with 2-D operations. Technical Report CG, Institute of Computer Graphics, Vienna University of Technology (1995)
14. Gobbetti, E., Pili, P., Zorcolo, A., Tuveri, M.: Interactive virtual angioscopy. In: Proceedings IEEE Visualization, Conference held in Research Triangle Park, NC, USA, IEEE Computer Society Press (1998) 435–438
15. Agus, M., Giachetti, A., Gobbetti, E., Zanetti, G., Zorcolo, A.: Real-time haptic and visual simulation of bone dissection. Presence: Teleoperators and Virtual Environments **12** (2003) 110–122
16. Agus, M., Giachetti, A., Gobbetti, E., Zanetti, G., Zorcolo, A.: A multiprocessor decoupled system for the simulation of temporal bone surgery. Computing and Visualization in Science **5** (2002)
17. Agus, M., Giachetti, A., Gobbetti, E., Zanetti, G., John, N.W., Stone, R.J.: Mastoidectomy simulation with combined visual and haptic feedback. In Westwood, J.D., Hoffmann, H.M., Mogel, G.T., Stredney, D., eds.: Medicine Meets Virtual Reality 2002, IOS Press (2002) 17–23

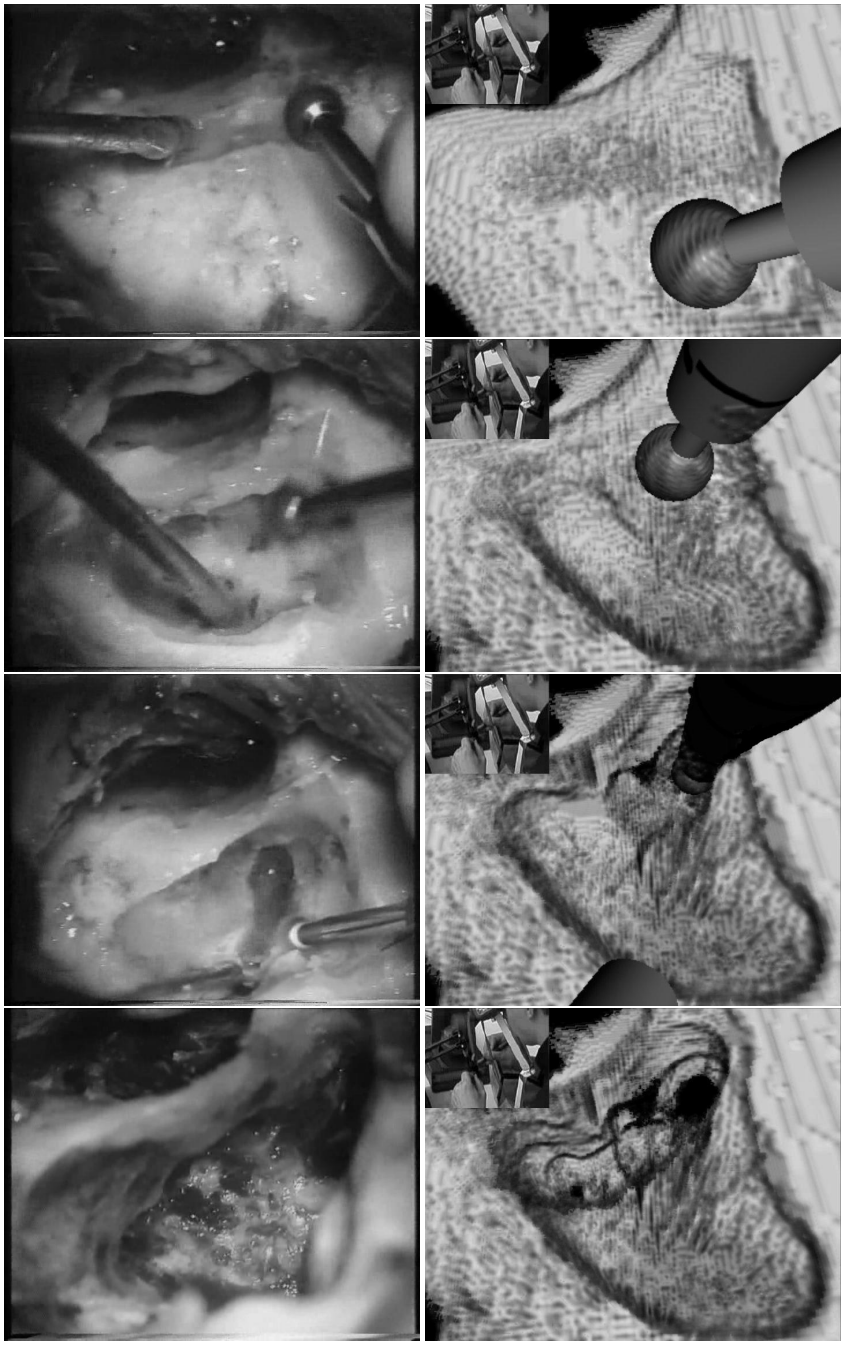

Fig. 2. Comparison between real and virtual intervention: the principal steps of a basic mastoidectomy, performed by a surgeon, are represented. Photos courtesy of Prof. Stefano Sellari Franceschini, University of Pisa.